CERTIFICATE OF MAILING UNDER 37 CFR§ 1.10

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail in an envelope addressed to: Assistant Commissioner of Patents, Washington, DC 20231 on June 29, 2000

EXPRESS MAIL LABEL: EJ207756395US

Amirah Scarborough
Name of Person Mailing Document

Amirah Scarborough
Signature of Person Mailing Document

# METHODS, SYSTEMS, AND COMPUTER PROGRAM PRODUCTS FOR DEFERRED COMPUTER PROGRAM TRACING

## Field of the Invention

The present invention relates generally to the field of computer programming, and, more particularly, to debugging of computer programs.

## Background of the Invention

Providing a software debug capability may be problematic at various phases during the software development life cycle. For example, operating systems may provide source code level debuggers that may be useful for initial design unit level testing on a host system. As the software product progresses through the development process, however, debuggers may not be available for target system testing and/or field testing. Some embedded software systems may not have the support of a software debugger or analyzer. Moreover, certain types of real-time sensitive software routines, such as device drivers, may not be compatible with conventional software debug tools.

One option for collecting debug data when testing high-level software is to call a conventional print function, which is typically provided in most high level programming languages. For example, the C programming language provides the printf() library function for writing data to a standard output. Unfortunately, the printf() function may be relatively inefficient, as parsing and formatting the arguments to the printf() function may be time consuming. As a result, real-time sensitive software routines, such as device drivers, may not be allowed to make printf() function calls. Furthermore, the inefficiency of the printf() function may alter a test scenario so drastically that it may become invalid.

Another option for collecting debug data is to implement a custom trace. A custom trace typically involves the definition of a set of symbols and the allocation of a trace buffer. The software may then be modified to write specific symbols into the

trace buffer when certain portions of code are executed. When a software error occurs, the trace buffer may be read and the symbols stored therein decoded. These symbols may then be used to determine the program flow that led to the error. While a custom trace may be more efficient than calls to a conventional print function, it may be more difficult to implement and maintain.

## Summary of the Invention

According to embodiments of the present invention, an application (*e.g.*, a computer program) prints data by invoking a print function with a format argument and, optionally, at least one data argument. The format argument and any data arguments are saved in a deferred trace data buffer. The print function returns to the application then, sometime after the print function has returned, the deferred trace data buffer is processed and the format argument and/or any data arguments are printed. By saving the format argument and any data argument(s) to a memory buffer instead of parsing and formatting the arguments in real-time, program efficiency may be improved and the impact of the print function on a software test scenario may be reduced.

In accordance with embodiments of the present invention, the format argument and any data argument(s) may be printed by retrieving the format argument and any data argument(s) from the deferred trace data buffer. Any data argument(s) retrieved may then be formatted based on the format argument and then printed. Advantageously, the format argument may be saved as a pointer in the deferred trace data buffer. As a result, the benefits of a conventional print routine may be provided while conserving memory in the deferred trace data buffer.

In accordance with other embodiments of the present invention, the format argument may be saved as a character string in the deferred trace buffer. This may be desirable if the address space of the application is no longer valid at the time the deferred trace data buffer is processed.

In accordance with still other embodiments of the present invention, if the format argument specifies a character string conversion, then the address of the corresponding data argument may be printed. In this case, the argument to be formatted is a pointer to a character string, which may not be relevant at the time the data arguments saved in the deferred trace data buffer are retrieved and formatted as

another process may have acquired the address space where the character string is stored in memory and may have overwritten the character string data.

In accordance with further embodiments of the present invention, a deferred print flag may be used to control whether the print function parses and formats its format argument and any data argument(s) in real-time or whether a deferred trace data buffer is created for saving the format argument and any data argument(s) so that they may be parsed and formatted at a later time.

In accordance with still further embodiments of the present invention, the print function that is invoked by the application and a trace data postprocessor function that may be used to print the format argument and/or any data argument(s) saved in the deferred trace data buffer may execute in different execution threads.

In accordance with still further embodiments of the present invention, the print function that is invoked by the application and a trace data postprocessor function that may be used to print the format argument and/or any data argument(s) saved in the deferred trace data buffer may execute on different computing machines.

Thus, the present invention may be used to provide a deferred print capability that may be useful in debugging software. For example, the present invention may be particularly useful in debugging software not supported by a commercial debug system or that may be adversely impacted by calls to conventional print library functions due to inefficiencies in parsing and formatting the data to be printed.

## Brief Description of the Drawings

Other features of the present invention will be more readily understood from the following detailed description of specific embodiments thereof when read in conjunction with the accompanying drawings, in which:

**FIG. 1** is a block diagram that illustrates data processing systems in accordance with embodiments of the present invention

**FIG. 2** is a block diagram that illustrates a processor and a memory that is configured with source code for a deferred print function and a trace data postprocessor function in accordance with embodiments of the present invention;

**FIG. 3** is a block diagram that illustrates a processor and a memory that is configured with a deferred print program and a trace data postprocessor program in accordance with embodiments of the present invention;

**FIG. 4** is a flowchart that illustrates exemplary operations of a deferred print program in accordance with embodiments of the present invention;

**FIG. 5** is a flowchart that illustrates exemplary operations of a trace data postprocessor program in accordance with embodiments of the present invention; and

**FIG. 6** is a diagram that illustrates exemplary embodiments of the present invention in which the deferred print program and the trace data postprocessor program execute on different data processing systems.

## Detailed Description of the Preferred Embodiments

The present invention will now be described more fully hereinafter with reference to the accompanying drawings, in which preferred embodiments of the invention are shown. This invention may, however, be embodied in different forms and should not be construed as limited to the embodiments set forth herein. Rather, these embodiments are provided so that this disclosure will be thorough and complete, and will fully convey the scope of the invention to those skilled in the art. Like reference numbers signify like elements throughout the description of the figures.

The present invention is described herein in the context of providing deferred computer program tracing using a modified version of the C/C++ programming language printf() library function. It will be understood that the present invention is not limited to the C or C++ programming languages as the principles and concepts of deferred computer program tracing discussed herein may be embodied in alternative programming languages.

The present invention may be embodied as methods, systems, and/or computer program products. Accordingly, the present invention may be embodied in software (including firmware, resident software, micro-code, *etc.*). Furthermore, the present invention may take the form of a computer program product on a computer-usable or computer-readable storage medium having computer-usable or computer readable program code embodied in the medium for use by or in connection with an instruction execution system. In the context of this document, a computer-usable or computer-readable medium may be any medium that can contain, store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus, or device.

The computer-usable or computer-readable medium may be, for example but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, device, or propagation medium. More specific examples (a nonexhaustive list) of the computer usable or computer-readable medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, and a portable compact disc read-only memory (CD-ROM). Note that the computer-usable or computer-readable medium could even be paper or another suitable medium upon which the program is printed, as the program can be electronically captured, via, for instance, optical scanning of the paper or other medium, then compiled, interpreted or otherwise processed in a suitable manner if necessary, and then stored in a computer memory.

Referring now to **FIG. 1**, an exemplary data processing system **10** in accordance with embodiments of the present invention typically includes input device(s) **12**, such as a keyboard or keypad, a display **13**, and a memory **14** that communicate with a processor **16**. The data processing system **10** may further include a storage system **18**, a speaker **19**, and an I/O data port(s) **20** that also communicate with the processor **16**. The storage system **18** may include removable and/or fixed media, such as floppy disks, ZIP drives, hard disks, or the like as well as virtual storage such as a RAMDISK. The I/O data port(s) **20** may be used to transfer information between the data processing system **10** and another computer system or a network (*e.g.*, the Internet). These components may be conventional components such as those used in many conventional computing devices, which may be configured to operate as described herein.

**FIG. 2** illustrates a processor **22** and a memory **24** in accordance with embodiments of methods, systems, and computer program products for deferred computer program tracing of the present invention and may be used in embodiments of data processing systems of **FIG. 1**. The processor **22** communicates with the memory **24** via an address/data bus **26.** The processor **22** may be, for example, a commercially available or customer microprocessor. The memory **24** is representative of the overall hierarchy of memory devices containing the software and data used to implement deferred computer program tracing in accordance with the

present invention. The memory may include, but is not limited to, the following types of devices: cache, ROM, PROM, EPROM, EEPROM, flash, SRAM, and DRAM.

As shown in **FIG. 2**, the memory **24** may hold four major categories of software and data: the operating system **28**, the source files module **32**, the library files/object module **34**, and the compiler module **36**. The operating system **28** controls the operation of the computer system. In particular, the operating system **28** may manage the computer system's resources and may coordinate execution of programs by the processor **22**. The source files module **32** includes those program files that may be compiled by the compiler module **36** into object code. Thus, the programs that are included in the source files module **32** may be called source programs because they represent the original form of the program as expressed in a particular programming language (*e.g.*, the C programming language).

In accordance with embodiments of the present invention, the source files module **32** includes a trace data postprocessor function **37**. Specifically, the trace data postprocessor function may be used to process and print data that has been stored in memory during execution of a computer program. Operations of the trace data postprocessor function **37** will be described in greater detail hereinafter.

In addition to the source files module **32**, a library files/objects module **34** may also be included, which comprises standard routines and utilities typically provided by the programming language. Many high level programming languages include a print function as part of their library. In accordance with particular embodiments of the present invention, a conventional C/C++ printf() library function may be modified to create a deferred printf() library function **38**. Unlike the conventional printf() library function, the deferred printf() library function **38** does not parse and format its arguments in real-time during execution, but instead saves its arguments in a buffer for later processing by the trace data postprocessor function **37**. Operations of the printf() library function **38** will be described in greater detail hereinafter.

The deferred printf() library function **38** along with any other library files from the library files/objects module **34** and source files from the source files module **32** may be compiled by the compiler module **36**. After the compiler module **36** has translated the source files and library files into an equivalent assembly language program, an assembler **42** may be invoked to translate the assembly language instructions into actual machine instructions that are understood by the processor **22**.

The machine instructions generated by the assembler **42** are often referred to as object code. The compiler module **36** may also include a linker **44**, which is used to link already compiled/assembled objects from the library files/objects module **34** with the object(s) generated based on the source files and library files. The linker **44** links all of the object code together to create an executable object. The executable object may be run or executed on the processor **22** or on another processor of the same type or that is compatible with the processor **22**. Alternatively, the compiler module **36** may comprise a cross-compiler, which may be used to create an executable object for execution on a target processor different from and/or incompatible with the processor **22**.

**FIG. 3** illustrates a processor **52** and a memory **54** that may be used to run or execute an executable object generated by the compiler module **36** of **FIG. 2** in accordance with embodiments of the present invention. As discussed in the foregoing, the processor **52** may not be compatible with the processor **22**. In this case, the compiler module **36** performs a cross compilation to generate an executable object for the processor **52**. On the other hand, the processor **52** and the memory **54** may represent the processor **22** and the memory **24** if the executable object generated by the compiler module **36** is intended to run on the same computer system. Finally, the processor **52** may be different from, but compatible with the processor **42**. The processor **52** communicates with the memory **54** via an address/data bus **56**. Similar to the processor **22**, the processor **52** may be, for example, a commercially available or customer microprocessor. The memory **54** is representative of the overall hierarchy of memory devices containing the software and data used to implement deferred computer program tracing in accordance with the present invention. The memory may include, but is not limited to, the following types of devices: cache, ROM, PROM, EPROM, EEPROM, flash, SRAM, and DRAM.

As shown in **FIG. 3**, the memory **54** may hold four major categories of software programs and data: the operating system **58**, the computer program(s) **62**, the trace data processor program **64**, and the deferred trace data buffer **66**. The role of the operating system **58** may be substantially identical to that of the operating system **28** discussed hereinabove with reference to **FIG. 2**. The computer program(s) **62** represent those computer program(s) that have been compiled for execution by the compiler module **36** of **FIG. 2**. These programs may include application programs,

driver programs, utility programs (*e.g.*, interrupt service routines, memory management routines, *etc.*) and the like. As used herein, the terms "application" or "application program" are used generically to encompass any program or executable code, including, for example, the operating system **58**, driver programs, utility programs, and traditional application programs (*i.e.*, programs designed to assist in the performance of a specific task).

A deferred trace program or deferred printf() library function program **68** is included with the computer program(s) **62** as the computer program(s) **62**, the operating system **58**, or other executable code (*i.e.*, any application program) may include functions and/or methods that may call the deferred printf() library function program **68** for debugging purposes. Specifically, the deferred printf() library function program **68** does not parse and format its arguments, including a format argument and one or more data arguments, in real-time during execution, but instead saves its arguments in the deferred trace data buffer **66** for later processing. The trace data processor program **64** includes the trace data postprocessor function **37**, which may be used to process and print data that has been stored in the deferred trace data buffer **66** by the deferred printf() library function **38** during execution of the computer program(s) **62** or other executable code.

Computer program code for carrying out operations of the present invention may be written in a high level programming language such as C or C++. In a preferred embodiment, the present invention uses the C programming language to implement most software programs. Nevertheless, it should be understood that the principles and concepts disclosed herein may also be applied to other programming languages. It should be further understood that the present invention may execute entirely on a single data processing system, comprising one or more processors, or may also be split between two different computing machines. For example, the deferred trace program **68** may execute on a first computing machine and the trace data postprocessor program **64** may execute on a second computing machine, different from the first computing machine, as will be discussed hereinafter.

The present invention is described hereinafter with reference to flowchart and/or block diagram illustrations of methods, systems, and computer program products according to an embodiment of the invention. It will be understood that each block of the flowchart and/or block diagram illustrations, and combinations of blocks

in the flowchart and/or block diagram illustrations, may be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions specified in the flowchart and/or block diagram block or blocks.

These computer program instructions may also be stored in a computer-usable or computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-usable or computer-readable memory produce an article of manufacture including instruction means that implement the function specified in the flowchart and/or block diagram block or blocks.

The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer implemented process such that the instructions that execute on the computer or other programmable apparatus provide steps for implementing the functions specified in the flowchart and/or block diagram block or blocks.

With reference to **FIGS. 4** and **5**, exemplary operations of methods, systems, and computer program products for deferred computer program tracing, in accordance with embodiments of the present invention, will be described hereafter. Referring now to **FIG. 4**, operations begin at block **82** where an application program (*i.e.*, any program or executable code, including, for example, the operating system **58**, driver programs, utility programs, and traditional application programs) executes on the processor **52**. At block **84**, a flag may be optionally set to control whether the printf() function will operate in a conventional manner or whether the printf() function will implement the deferred print capability of the present invention. For example, a software developer may place printf() statements in various portions of code to print out data during software testing. The software developer may then set the deferred printf() flag to ensure that calls to the printf() function result in a deferred print operation rather than a real-time print operation, which could invalidate the test results

or create additional errors if executed in a real-time sensitive portion of code (*e.g.*, in a driver function).

At block **86**, an application program function or method calls the deferred printf() library function program **68** to perform a print operation. In an exemplary embodiment, the deferred printf() library function **38** is implemented by modifying the source code of the conventional printf() library function. The general format of a printf() call is as follows:

printf(format, arg1, arg2, ...)

The format argument is an address of a character string that describes how the remaining data arguments (*i.e.*, arg1, arg2, ...) are to be displayed. The characters inside the format string that are not preceded by % signs are written, without any conversion, to the standard output. Otherwise, the % sign will be followed by one or more characters that specify the format for displaying the corresponding argument to printf(). The % sign and the format characters that follow it are often referred to as conversion characters.

As discussed hereinabove, a drawback to the use of the conventional printf() library function for performing software program tracing/debugging is the inefficiency in parsing the arguments and formatting the output based on the format argument. Advantageously, in accordance with the present invention, the deferred printf() library function program **68** makes a determination at block **88** whether the flag has been set for deferred printing at block **84.** If the flag has not been set, then the deferred printf() library function program **68** performs a conventional printf() print operation at block **92**. That is, the arguments to the deferred printf() function are parsed, formatted, and printed to the standard output in real-time. Note that by using a flag to control whether a deferred print operation is to be performed allows the deferred printf() function **38** to be implemented through a modification to the conventional printf() library function. This may allow software developers to use one type of function call to perform a print operation rather than having separate deferred print and conventional print functions, which may result in more complicated code.

Nevertheless, it should be understood that, in accordance with the present invention, the deferred printf() library function program **68** may be embodied as a separate routine or program from the conventional printf() library function. In this embodiment, a flag for controlling the operations of the conventional printf() library

function would be unnecessary as control over whether a deferred or real-time print operation is performed would be through selection of either the conventional printf() library function or the deferred printf() library function program **68** in the application program code.

If the flag has been set for deferred printing as determined at block **88**, then, at block **94**, the deferred printf() library function program **68** may create the deferred trace data buffer **66** in the memory **54.** The deferred printf() library function program **68** then stores the arguments to the deferred printf() function in the deferred trace data buffer **66** in real-time at block **96** without performing any parsing or formatting operations on the arguments and returns to the calling application function or method. The arguments may include a format argument, which is a pointer to a format string, and one or more optional data arguments. The deferred printf() library function program **68** may be more efficient than the conventional printf() library function program because the argument parsing and formatting operations are postponed to a later time. As a result, an application program may call the deferred printf() library function program **68** without adversely affecting the integrity of a test scenario.

The deferred printf() library function program **68** may store the format argument as a pointer in the deferred trace data buffer **66** rather than saving the contents of the address space of the application that is referenced by the format argument pointer (*i.e.*, the format data string). As a result, this may allow for reduced memory consumption in the deferred trace data buffer **66**. Alternatively, if the format argument pointer will not be valid when the deferred trace data buffer **66** is processed at a later time, then the contents of the memory location in the address space of the application that is referenced by the format argument pointer (*i.e.*, the format data string) may be saved in the deferred trace data buffer **66**. In this case, more memory may be consumed in the deferred trace data buffer **66**, but the deferred trace data buffer **66** may still be processed even if the application address space is no longer valid or cannot be trusted (*e.g.*, the address space has been overwritten or given up to another application).

Finally, as represented by the connector **A**, the foregoing operations continue indefinitely for as long as the application program executes on the processor **52**. Note that the operation of creating the deferred trace data buffer **66** at block **94** may be performed only if the deferred trace data buffer **66** has not been created previously by

a call to the deferred printf() library function program **68**. If the deferred trace data buffer **66** has already been created, then the deferred printf() library function program **68** may obtain a pointer to the deferred trace data buffer **66** in memory at block **94**, which may be used to access the deferred trace data buffer **66** at block **96**. Note that the trace data buffer **66** may alternatively be created by another program at initialization, for example, thereby obviating the need for the deferred printf() library function program **68** to create the deferred trace data buffer **66**.

Referring now to **FIG. 5**, the exemplary operations of the trace data postprocessor program **64** in processing the data stored in the deferred trace data buffer **66** will be described hereafter. The trace data postprocessor program **64** may execute as part of the same execution thread in which the deferred printf() library function program **68** executes. Typically, however, the trace data postprocessor program **64** executes as part of a different execution thread from that in which the deferred printf() library function program **68** executes. For example, the trace data postprocessor program **64** may be executed as part of a thread that is run periodically or after a test scenario has executed to print out the program trace data that has been collected.

Operations begin at block **102** where the trace data postprocessor program **64** executes on the processor **52**. At block **104**, the trace data postprocessor program **64** retrieves the trace data (*i.e.*, the arguments from a deferred printf() library function program **68** call) from the deferred trace data buffer **66**. As discussed hereinabove, these arguments include the format argument and, optionally, one or more data arguments. The trace data postprocessor program **64** may then format and print the arguments to the deferred printf() library function program **68** at block **106** using the format argument, which was passed as the first argument. Advantageously, the benefits of a conventional printf() function call may be provided. Recall, however, that the format argument is a pointer to a character string. Thus, to format the other data arguments passed to the deferred printf() library function program **68** correctly, the character string is preferably stored in static memory or precautions should be taken to ensure that the character string is not moved or overwritten in the memory **54** until after the trace data postprocessor program **64** processes the format argument in the deferred trace data buffer **66**. As discussed hereinabove, if the format argument pointer will be invalid at the time the trace data postprocessor program **64** processes

the deferred trace data buffer **66**, then the format character string itself, rather than the format argument pointer, should be stored in the deferred trace data buffer **66**.

At block **108**, the trace data postprocessor program **64** checks to determine whether the format argument specifies that one of the other arguments is to be formatted as a character string (*i.e.*, a %s is encountered in the format argument). In this case, the argument to be formatted is a pointer to the character string, which may not be relevant at the time the trace data postprocessor program **64** executes as another process may have acquired the address space in the memory **54** where the character string is stored and may have overwritten the character string data. Therefore, in a preferred embodiment of the present invention, the trace data postprocessor program **64** prints the address where the character string resided when the deferred printf() library function program **68** was called at block **112**. Finally, as represented by block **114**, the trace data postprocessor program **64** continues to process all the arguments stored in the deferred trace data buffer **66** for a particular call to the deferred printf() library function program **68** until all the arguments have been processed.

**FIG. 6** illustrates alternative embodiments of the present invention in which the deferred printf() library function program **68** and the trace data postprocessor program **64** execute on different computing machines. Referring now to **FIG. 6**, the deferred printf() library function program **68** may execute on a first data processing system **10a**, which has access to a non-volatile storage medium **122**. The operations described hereinabove with respect to **FIG. 4** may be performed on the first data processing system **10a**. Next, the deferred trace data buffer and the application program address space may be saved in the non-volatile storage medium **122**. The contents of the non-volatile storage medium **122** may then be provided to a second data processing system **10b** as shown in **FIG. 6** where the trace data postprocessor program **64** performs the operations described hereinabove with respect to **FIG. 5**. Because the application program address space is saved along with the deferred trace data buffer **66** in the non-volatile storage medium **122**, the format argument pointer(s) saved in the deferred trace data buffer **66** may still be interpreted. If, however, the format character string itself, rather than the format argument pointer, has been stored in the deferred trace data buffer **66**, then the application program address space need not be saved in the non-volatile storage medium.

The flow charts of **FIGS. 4** and **5** illustrate the architecture, functionality, and operations of a possible implementation of the trace data postprocessor function **37** and deferred printf() library function **38** software. In this regard, each block represents a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that in some alternative implementations, the functions noted in the blocks may occur out of the order noted in **FIGS. 4** and **5**. For example, two blocks shown in succession may in fact be executed substantially concurrently or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved.

From the foregoing it can readily be seen that, in accordance with the present invention, software programs may be debugged using deferred computer program tracing methods, systems, and computer program products. Specifically, a computer program may include calls to a deferred printf() library function program that will log its arguments in a buffer in memory for parsing and formatting at a later time. By eliminating the parsing and formatting of the function arguments in real-time, program efficiency may be improved and the likelihood of impacting a software test scenario may be reduced. A trace data postprocessor program may then be used to retrieve the arguments stored in the memory buffer where they are then parsed, formatted, and printed. Advantageously, because the format argument may be saved in the memory buffer as a pointer, memory resources may be conserved while still providing the benefits of a conventional printf() function call.

In concluding the detailed description, it should be noted that many variations and modifications can be made to the preferred embodiments without substantially departing from the principles of the present invention. All such variations and modifications are intended to be included herein within the scope of the present invention, as set forth in the following claims.